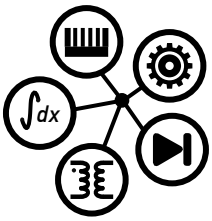

PLECS

DEMO MODEL

Buck Converter with Parameter Sweep

Using the parallel simulation feature in PLECS Standalone

Last updated in PLECS 4.8.1

1 Overview

This demonstration is based on the demo model "Buck Converter with Voltage Controls" in the PLECS demo models library. It performs a parameter sweep by modifying the value of inductor L_1 in a simulation script. As of version 4.6.1 PLECS Standalone features the possibility to perform parallel simulations, which is also demonstrated in this demo model.

1.1 Requirements/Limitations

Different PLECS features are shown in this demo model. Please note that you can only use the features for your version of PLECS (Blockset or Standalone).

- The PLECS Blockset model features a simulation script with a sequential parameter sweep.
- The PLECS Standalone model features simulation scripts for both sequential and parallel implementations of the parameter sweep.
- To run the Python script for the parallel parameter sweep a Python 3 installation and PLECS Standalone is necessary. Also, the RPC interface needs to be enabled and configured for port 1080 under the **PLECS Preferences + General** tab.

2 Model

The schematic shows a buck converter with an analog proportional integral derivative (PID) controller. In this example the inductor L_1 is labeled in the schematic in Fig. 1 and will be simulated ten times in an automated fashion, with a value from 40 to $220\,\mu$H in steps of $20\,\mu$H. The Signal Outport connected to the Probe component hands data to the simulation script and the RPC interface.

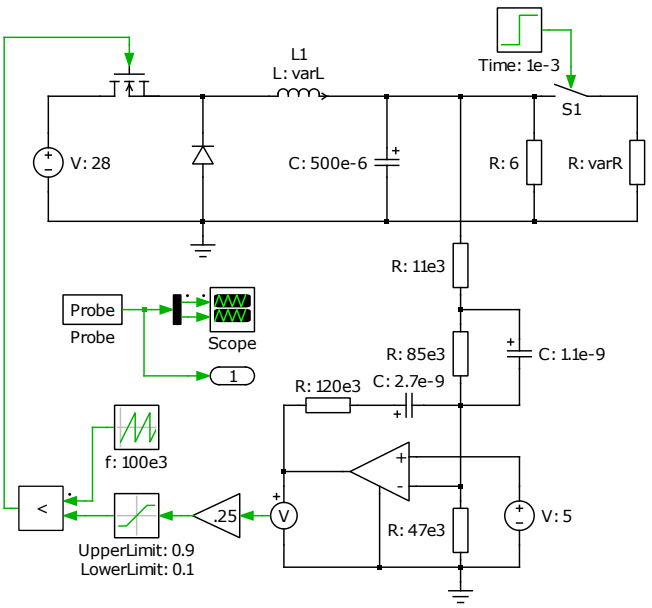

Figure 1: Schematic of the buck converter with analog controls. Note the Signal Outport connected to the Probe component which is used hand back data to the simulation scripts.

3 Simulation

The simulation demonstrates the start-up of the converter and a load jump at $1\,\mathrm{s}$. A single transient simulation can be performed where an inductance value of $40\,\mu\mathrm{H}$ is assigned to L_1 from the model initializations. Alternatively a parameter sweep can be performed for L_1 by using the simulation script defined under **Simulation + Simulation Scripts**. The result of each simulation is displayed as a new trace in the scope. Each trace is labeled with the corresponding inductance value. The script also analyzes the simulation result and prints the peak current value into the MATLAB or Octave console.

3.1 Octave Script for PLECS Standalone

There are two simulation scripts that implement a parameter sweep for the inductor values of the Buck converter. One performs a sequential and the other a parallel parameter sweep.

Parameter Sweep (Sequential)

The sequential parameter sweep runs several consecutive simulations in a for-loop. Only when a simulation is finished the next simulation will start. In each iteration of the for loop the `ModelVars` struct is assigned a new value for the inductor (variable `varL`). The `ModelVars` struct is part of the `simStruct` which is handed over to PLECS as a parameter of the `plecs('simulate')` command. The `SolverOpts` struct contains a variable `OutputTimes` which returns simulation data only for specific points in time. This allows one to reduce the amount of data that is processed inside the simulation script (see section 3.2). In this demo model, with the default solver settings and the selected `OutputTimes` vector, the output simulation data is reduced from approximately 4300 to 501 points.

```
% create simStruct with field 'ModelVars'
mdlVars = struct('varL', 50e-6);
simStruct = struct('ModelVars', mdlVars);

% clear all previous traces in scope 'Scope' in the current model
plecs('scope', './Scope', 'ClearTraces');

% parametric values to be swept
inductorValues = [40:20:220]; % in uH

for ix = 1:length(inductorValues)
   % set value for L1
   simStruct.ModelVars.varL = inductorValues(ix) * 1e-6;
   simStruct.SolverOpts.OutputTimes = 0.001:0.2e-5:0.002;
   % start simulation, return probed signal values in 'out'
   out = plecs('simulate', simStruct);
   % hold and label trace
   plecs('scope', './Scope', 'HoldTrace', ['L=' mat2str(inductorValues(ix)) 'uH']);
   % find maximum current value and index
   [maxv, maxidx] = max(out.Values(1,:));
   % Output maximum current values to Octave console
   printf('Max current for L=%duH: %fA at %fs\n',
   inductorValues(ix), maxv, out.Time(maxidx));
end
```

Parameter Sweep (Parallel)

To run a parallel simulation not only one simulation struct but multiple need to be specified as a cell array for the parameter of the `plecs('simulate')` command. Each simulation struct may contain an additional variable called `Name`. This allows to label each individual parallel simulation and to associate

simulation data or errors to a particular simulation. In this example an artificial short circuit is created when $L_1 = 120\,\mu$H (simulation 5). The error message is displayed in the Octave console and in the bottom right corner in the PLECS schematic after the simulation has been completed, see Fig. 2.

Figure 2: Error message symbol for simulation 5 that shows a short circuit at the load.

There is an additional parameter in the `plecs('simulate')` command for the parallel simulation setup. It is a callback function which is executed after each simulation. The function can be used to reduce the amount of simulation data to only a few interesting key figures. This is especially useful when starting a simulation through the RPC interface, since avoiding the transfer of large amounts of data is desired.

```
% clear all previous traces in scope 'Scope' in the current model
plecs('clc');
plecs('scope', './Scope', 'ClearTraces');

% Evaluate simulation results in callback function
function result = callback(index, data)
   % hold and label trace
   name = ['L = ', mat2str(inductorValues(index)), 'uH'];
   plecs('scope', './Scope', 'HoldTrace', name);

   % Find maximum current values and index
   if isstruct(data)
      [maxi, maxidx] = max(data.Values(1,:));
      maxt = data.Time(maxidx);
      % Reducing simulation results by return value 'result'
      result = [maxi, maxt];
   else
      % Print error message to Octave console
      printf(' Error in Simulation %d for L=%duH: %s\n',
                  index, inductorValues(index), data);
   end
end

% Set value for L1 to be swept
inductorValues = [40:20:220]; % in uH
% Initialize simStruct as cell array with all values for L1
for ix = 1:length(inductorValues)
   simStructs{ix}.ModelVars.varL = inductorValues(ix) * 1e-6;
   % Name of 'ModelVars' can be assigned for diagnostic purposes
   simStructs{ix}.Name = ['L=' mat2str(inductorValues(ix)) 'uH'];
end

% Create a shortcut in simulation 5
simStructs{5}.ModelVars.varR = 0;

% Start simulation, return result from callback function into 'out'
% Analysis will be moved to callback function to reduce simulation results
out = plecs('simulate', simStructs, @(index, data) callback(index, data));

for ix = 1:length(inductorValues)
   % Detect if errors occurred in parallel simulation
   if ischar(out{ix})
      printf(' Error for L=%duH: %s\n',
      inductorValues(ix), out{ix});
      % Output maximum current values to Octave console
```

```
    else
       printf(' Max current for L=%duH: %fA at %fs\n',
       inductorValues(ix), out{ix}(1), out{ix}(2));
    end
end
```

The same parallel simulation can also be started by using Python 3 and the provided python script `parameter_sweep_script.py` in the folder of this demo model. Please note that this needs the RPC interface to be enabled on port 1080 in the **PLECS Preferences** menu. The simulation results for the parameter sweep are are given in Fig. 3.

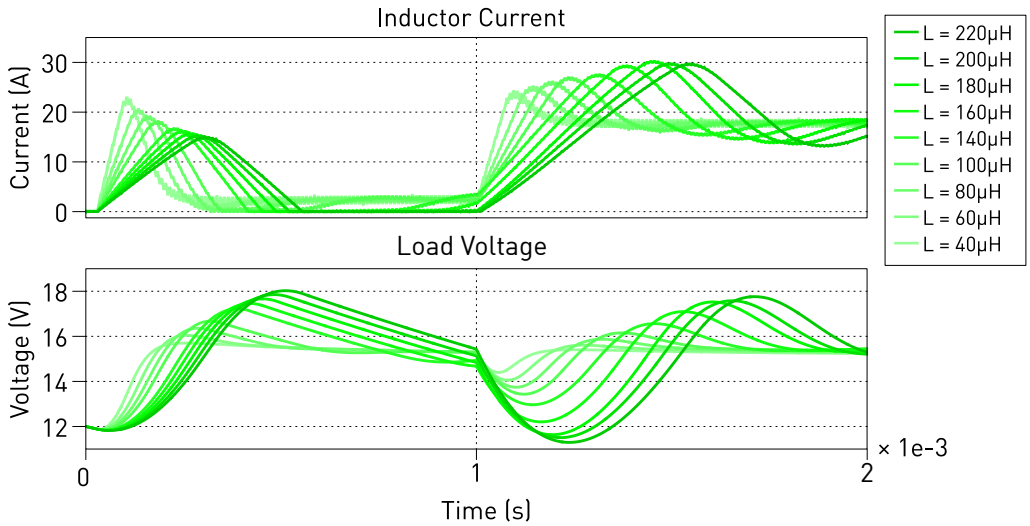

Figure 3: Output results of the parameter sweep

3.2 Reduce the Amount of Simulation Data

Depending on the simulation time-span, the average simulation time-step and the number of recorded signals the resulting amount of simulation data can be substantial. With simulations running in parallel the memory consumption further grows. The following points present ways to minimize the amount simulation data.

- The **Max step size** parameter in the **Simulation Parameters** dialog should not be decreased.
- The evaluation of simulation results should be done in the corresponding callback function as shown in the script `Parameter Sweep (Parallel)`. This way only the relevant simulation data (`result`) is returned to the script. If no callback function for post processing is used, all simulation results are handed back to the simulation script (return value `out`).
- A time vector `OutputTimes` with an arbitrary step size within the simulation time can be specified to reduce simulation data. In combination with `SolverOpts` a struct variable that allows you to override the solver settings specified in the Simulation Parameters dialog must be used. This is demonstrated in the Parameter Sweep (Sequential) in line 14.

The results are printed into the Octave console which can be accessed by selecting **Show Console** from the **Window** menu.

3.3 Python 3 Script for PLECS Standalone

The same simulation scripts, as described in the previous section, are implemented in Python 3. The scripts can be run with the following command:

```
python3 parameter_sweep_script.py
```

To select the simulation type (e.g., sequential or parallel), please adjust the variable `SIMULATION_TYPE` within the Python script, accordingly. Please make sure to enable the RPC interface under the **PLECS Preferences + General** tab and set the port to `1080`. To execute the script PLECS must be first started manually.

3.4 Script for PLECS Blockset

Double-click on the subsystem block in the Simulink model to view and run the m-file `parameter_sweep_script.m` in the MATLAB editor which has the following content:

```
% create path to scope
scope = ('buck_converter_with_parameter_sweep/Circuit/Scope');

% clear all previous traces in scope 'Scope' in the current model
plecs('scope', scope, 'ClearTraces');

% parametric values to be swept
inductorValues = [40:20:220]; % in uH

for ix = 1:length(inductorValues)
   % set value for L1
   varL = inductorValues(ix) * 1e-6;
   % start simulation, return probed signal values
   % to workspace using Output port '1'
   [t, x, y] = sim('buck_converter_with_parameter_sweep');
   % hold and label traces in scope
   plecs('scope', scope, 'HoldTrace', ['L=' mat2str(inductorValues(ix)) 'uH']);
   % find maximum current value and index
   [maxv, maxidx] = max(y(:,1));
   % Output maximum current values to MATLAB console
   fprintf('Max current for L=%duH: %fA at %fs\n',
      inductorValues(ix), maxv, t(maxidx));
end
```

The results are printed into the MATLAB console.

4 Conclusion

This model demonstrates using a simulation script to perform a parameter sweep of a physical circuit value in either PLECS Blockset or PLECS Standalone. This example code can be readily be adapted to other applications.

Revision History:

PLECS 4.3.1	First release.
PLECS 4.6.1	Add parallel implementation of parameter sweep in PLECS Standalone. Add Python 3 script to run the parameter sweep over XML-RPC.
PLECS 4.6.5	Include the option to use JSON-RPC.
PLECS 4.8.1	Add sequential implementation of the parameter sweep in Python 3 script. Fixed minor issues with JSON-RPC.

How to Contact Plexim:

☎	+41 44 533 51 00	Phone
	+41 44 533 51 01	Fax
✉	Plexim GmbH Technoparkstrasse 1 8005 Zurich Switzerland	Mail
@	`info@plexim.com`	Email
	`http://www.plexim.com`	Web

PLECS Demo Model